On using generic profiles and views for dynamic web services adaptation

Sébastien Laborie, Bouchra Soukkarieh, Florence Sèdes

On Using Generic Profiles and Views for Dynamic Web Services Adaptation

Sébastien Laborie [*1], Bouchra Soukkarieh [2], Florence Sèdes [3]

IRIT, Université Paul Sabatier, 118 Route de Narbonne, F-31062 TOULOUSE

LIUPPA – T2i, Université de Pau et des Pays de l'Adour

[*1] sebastien.laborie@iutbayonne.univ-pau.fr ; [2] bouchra.soukkarieh@irit.fr; [3] florence.sedes@irit.fr

Abstract- **The emergence of mobile technologies allows users to be connected to services anytime, anywhere and anyhow. However, this mobility requires new constraints on the execution of the services and their presentations to users. Hence, these ones must be selected and adapted according to not only the user's profile but also his context.**

In this paper, we present a new context-aware adaptation architecture based on Web Services, named CA-WIS. Our proposal aims users to interact with web services that correspond to their needs and their contexts.

Keywords- Context-Aware Adaptation; Web Service

I. INTRODUCTION

Nowadays, users use various devices, such as laptops, smartphones, tablets, and several types of networks (remote and local) to access resources and services. These context varieties create an increasing need for users to execute dynamically services that correspond to their current context.

In order to ensure the usability of Web Services (WS), it is necessary to integrate into Web Service architectures context information. This context must describe information about the user's environment (device, network, location…) and his profile (preferences, static characteristics...).

Therefore, our objective is to propose a context-aware adaptation architecture based on Web Services. To realize this objective, we extended the classic Web Service architecture by adding an adaptation layer that contains various components dedicated to the management of the adaptation process, the user and the service contexts. Our architecture is named CA-WIS (Context Adaptation of Web Information System) platform. In this platform, we take into account the context in two phases:

- The research phase: select a list of relevant services that correspond to the user needs and context.
- The interaction phase: adapt dynamically the executed services to the evolution of the user environment.

This platform allows users on the one hand to receive a list of adaptive Web Services to his current context, and on the other hand to interact with web services that correspond to their needs and their current contexts. The adaptation process in this platform is based on a methodology based on data modeling [8]. In this methodology, Web Services are provided with specific profiles (i.e., service descriptions) and views (i.e. the service output rendering, such as a Web page). Then, based on the current user context, it selects the relevant service profiles and views that comply with the user's constraints. However, in such a framework, it might be possible that no specific view corresponds to the user context. Hence, in such a case, the user could not use the desired service. Furthermore, since each specific service profile is compared to the current user context, if many services are provided and if the user context evolves frequently, the system may be overloaded.

In order to overcome these limitations, we propose, in this paper, an enhancement of our framework that aims to group some specific profiles and views into generic profiles and views. This grouping accelerates the search process of relevant specific service profiles and views. Moreover, if not relevant specific view corresponds to the user's context, based on generic views, our method can generate new specific views. Consequently, our proposal ensures the usability of Web Services at anytime, anywhere and anyhow.

The remainder of this paper is structured as follows. In Section 2, some web service architectures related to our CA-WIS platform are presented. Section 3 describes our Web Service architecture which dynamically takes into account the user contexts. Section 4 details our proposed algorithms for constructing generic profiles and views. Section 5 illustrates our method that uses generic profiles for selecting specific service profiles and views. Section 6 describes how new generic views may be used to compute new specific views. Section 7 presents our prototype. Finally, Section 8 concludes the paper with brief concluding remarks.

II. RELATED WORK

In this section, we take a look at some frameworks that take into account the user context in order to select and adapt Web Services.

- [1] has proposed an architecture that aims to realize dynamic services adaptation according to the user's context. This architecture takes into account the user and services context. These contexts are handled by an "Adapter" component in this architecture to verify the compatibility between the user profile and the profile of each component of the service. When the Adapter detects the incompatibility between the profiles, it starts an adaptation process by adding, removing or replacing components of the service.

  Indeed, the main disadvantage of this architecture is its complexity.

- [2] has presented an architecture that aims to dynamically adapt services to the user's context where each service is composed of a set of interconnected components. Several versions of the components of each service are saved. The service adaptation aims to select the service components that correspond to the context.

  Indeed, the use of this architecture leads to a server overload.

- [3] has proposed a generic architecture in order to provide adaptable services to mobile users. A database of multimedia data and another database of services and/or services components are generated by the service provider. These two databases contain several versions of service data and components in order to select the relevant version of the service to user's context.

  The adaptation method used in this architecture causes the saturation of the system.

- [6] has proposed the CATIS system which has been evaluated in the tourism domain. This system adapts only the result of the services to the user's context. Moreover, all the results displayed are always presented in text mode.

Every framework presented above aims to save multiple versions of services components and/or data in order to realize the adaptation process. These methods used in the previous frameworks usually cause the slow down and the saturation of system. Moreover, it may happen that no service could be provided to the user even if he wants to access to a service at anytime. Therefore, we propose in this article a new method that aims to treat this problem and to propose some relevant solutions.

III. CA-WISPLATFORM

The architecture of our CA-WIS platform is based on the WS architecture (Figure 1) that is composed of three entities (provider, user, and registry) and is based on three standards (WSDL (Web Service Description Language) [11], UDDI (Universal Description, Discovery and integration) [10] and SOAP (Simple Object Access Protocol) [9]). The service provider builds the service and publishes its description in a registry. The user needs are translated into requests that are transmitted to the WS registry. Once the service is found, the user will obtain direct interaction with the service [5].

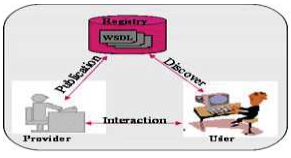


Fig. 1 Web Service architecture

In our CA-WIS architecture, we extend the WS architecture by adding an adaptation layer which contains various components dedicated to the context acquisition and adaptation. The context, in this architecture, is presented by a generic model regarding the user and the service (for more details, see [8] [7]).

Our architecture is thus composed of three layers depending on each other (Figure 2):

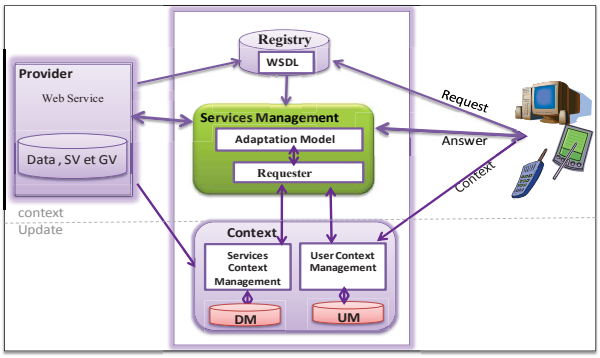


Fig. 2 CA-WS architecture

1) The **Registry layer** that corresponds to a registry of service description offering facilities that publish services for providers and that searches services for users. This layer doesn't play any role in the interaction phase; it is used exclusively during the research phase.

2) The **Services Management** layer is responsible for managing the adaptation process and computing the final results to the user. This fundamental layer is composed of two components.

i. The "Requester" that is responsible of verifying the user's context evolution by contacting the User Context Management component.

ii. The "Adaptation Model" that is responsible for the adaptation process.

3) The **Context layer** is responsible for capturing and managing the user and the service context. It is composed of two modules:

i. User Context Management that is responsible for capturing the user context and storing it in a database (User Model (UM)). The user context is obtained in an explicit and implicit way. More precisely, when the user launches his request, he may express explicitly his static characteristics and his preferences but remaining characteristics (localization, network, etc) are obtained implicitly.

ii. Services Context Management that is responsible for the extraction of the Web Services contexts, the storage of these contexts in a database (Domain Model (DM)). The service context is expressed directly by the service provider.

The search of the web services and the interaction between the user and the service is not direct but is realized by the **Services Management**. For each connection from the user to the services, the **Services Management**, thanks to its components, verifies the evolution of the user's context and the compatibility between the new user context and the service context. If the two contexts are not compatible, the **Services Management** triggers an adaptation process by the Adaptation Model component.

Our CA-WIS platform requests the Web Service provider to provide several views of its service data complying with different profiles. The adaptation process realized by the **Services Management** aims to find the relevant view that corresponds to the current user profile. If the Services Management doesn't find a relevant view, it can't realize an adaptation process (for more detail, see [7]).

To solve this problem, we propose, in this paper, a new method that aims (1) to accelerate the selection of specific service profiles and views by grouping the profiles and views provided by the service provider into generic profiles and views, and (2) to ensure the execution of services by generating specific views from generic views.

In the next paragraph, we detail how the **Services Management** uses generic profiles and generic views to adapt dynamically service to the user's context.

IV. CONSTRUCTING GENERIC PROFILES AND VIEWS

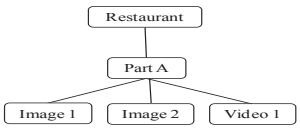
Web Service providers may produce several Web Service presentations complying with different profiles. For instance, Figures 3 and 4 illustrate two Web Services presentations which allow different kind of iPhone users to see particular information about a restaurant. Specific profiles (e.g., *Sp1* and *Sp2*) have been identified in order to execute the web services correctly on a target device. Moreover, specific views (e.g., *Sv1* and *Sv2*) complying the specific profiles have been designed accordingly.

In the following, we show how to construct, from specific profiles and views; generic profiles (§4.1) and views (§4.2).

(a) Screenshot.

C1: Required bandwidth > 4MO.
C2: Languages = {french}.
C3: Images are prefered.
C4: Video are prefered.

(b) Specific profile1 (*Sp1*).



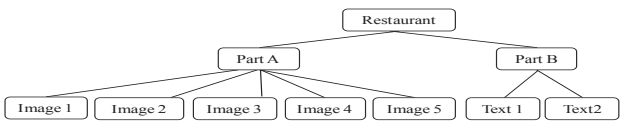
(C) Specific View 1 (*Sv1*).

Fig. 3 A specific presentation for iPhone

(a) Screenshot.

C1: Required bandwidth > 1MO.
C2: Languages = {english}.
C3: Images are prefered.
C4: Texts are prefered.

(b) Specific profile 2 (Sp2).



(C) Specific View 2 (Sv2).

Fig. 4 Another specific presentation for iPhone

A. Constructing Generic Profiles

A profile is a description of all the capabilities and limitations of a given terminal [4]. Usually, profiles contain the description of both the software and hardware features of a terminal. Software capabilities list, for example, the supported codecs, whether or not the operating system is capable of handling parallel videos or audio streams, etc. The hardware capabilities are given in terms of the physical display size, the input modalities, such as keyboard and pointers. Profiles may also capture user preferences. For example, the user may prefer a particular text magnification for readability purposes. In a more abstract way, a profile can be viewed as a set of constraints on presentations introduced by profile descriptions. Examples of profiles are illustrated in Figures 3(b) and 4(b). From such specific profiles, we propose to compute a generic profile combining all their constraints.

Definition 1 (**A generic profile**): A generic profile refers to a list of constraints and groups several specific contextual information into a global structure.

In order to construct a generic profile *Gp* from some specific profiles *Spi*, we have defined the following procedure:

- for each constraint *Ck* in *Spi*
  - if $Ck \notin Gp$, add *Ck* into *Gp*;
  - otherwise, merge *Ck* with its corresponding constraint *C* in *Gp*, e.g., add *Ck* in the set of constraints *C* or relax *C* in order to take into account *Ck*.

Example 1. Suppose that we want to construct a generic profile *Gp* from the specific profiles *Sp1* (Figure. 3(b)) and *Sp2* (Figure. 4(b)). Thanks to our above defined procedure, all preferred media are added into *Gp*. Moreover, the bandwidth constraints and the used languages are merged into *Gp*. Actually, the bandwidth should be superior to 1Mo for executing both presentations. Furthermore, the presentations may be played in French or in English. Figure 5 presents the overall generic profile *Gp* integrating all the specific contextual information of Figures 3(b) and 4(b).

c1: Required bandwidth > 1Mo.

c2: Languages = {French, English}.

c3: Images are preferred.

c4: Videos are preferred.

c5: Texts are preferred.

Fig. 5 A generic profile Gp constructed from Fig. 3(b) and Fig. 4(b)

In the next paragraph, we present how to construct from specific views, generic views and we show our objective of constructing generic profiles and views.

B. Constructing Generic Views

A view corresponds to a web service presentation structure. As done, for instance, for XHTML documents, a view can be composed of elements and sub-elements characterizing the organization of the presentation. The root element of a view refers to the whole presentation and each leaf corresponds to media content.

Examples of specific views are presented in Figure 3(c) and 4(c). From such views, we propose to compute a generic view combining all of their information.

Definition 2 (A generic view): A generic view is a structure which combines information about specific views. Moreover, each element of the generic view is associated to a value which corresponds to its importance. For instance, if an element appears many times in some specific views, this will increase the importance of the element inside the generic view.

Algorithm 1, named **ConstructGenericView**, constructs a generic view given different specific views.

Algorithm 1: ConstructGenericView

Input: One node *sn* related to a specific view and another node *gn* related to the generic view.

1 $L_{child_{sn}} \leftarrow child(sn)$;
2 $L_{child_{gn}} \leftarrow child(gn)$;
3 $L_1 \leftarrow L_{child_{sn}} \setminus L_{child_{gn}}$;
4 $L_2 \leftarrow identicalNodes(L_{child_{sn}}, L_{child_{gn}})$;
5 **for** *each element* $e_i \in L_1$ **do**
6 add the child e_i (with all of its descendants) to the node gn;
7 label all arcs between e_i and gn, and e_i descendants with the value $v = 1$;
8 **end**
9 **for** *each pair of elements* $\langle e_s, e_g \rangle \in L_2$ **do**
10 increment the value v on the arc between gn and e_g;
11 $ConstructGenericView(e_s, e_g)$;
12 **end**

Example 2. Suppose that we want to construct a generic view *Gv* from the specific views *Sv1* (Figure 3(c)) and *Sv2* (Figure 4(c)). Firstly, we call the algorithm *ConstructGenericView(root(Sv1), root(Gv))*. Actually, it copies *Sv1* into *Gv* and all arcs in *Gv* are labeled by the Value 1. Thereafter, we call the algorithm *ConstructGenericView (root(Sv2), root(Gv))* in order to update *Gv* with new information coming from other specific views, here *Sv2*. From lines 1 to 4, the algorithm initializes the following lists: Lchildsn = {PartA, PartB}, Lchildgn = {PartA}, L1 = {PartB} and L2 = {<PartA, PartA>} (The function *identicalNodes* returns a list of couples where its specific and generic elements are identical).

From L1, the algorithm adds into *Gv* the subtree corresponding to Part B and all arcs are labeled by 1 (Lines 5 to 8). From L2, we increment the importance of the PartA (it appears in *Sv1* and *Sv2*, hence v = 2) and the algorithm is calling recursively on the elements of the couple of nodes <PartA; PartA> (Lines 9 to 12). During this recursive call, the new values for the lists follow: Lchildsn = {Image1, Image2, Image3, Image4, Image5}, Lchildgn = {Image1, Image2, Video1}, L1 = {Image3, Image4, Image5} and L2 = {<Image1, Image1>, <Image2, Image2>}. From L1, its elements are added into the generic view *Gv*. From L2, we increment the importance of the Image1 and Image2, and the algorithm is calling recursively on its elements... This process is repeated until all elements in L2 have been treated, i.e., until new information have to be incorporated into *Gv*.

When the algorithm stops, it produces the generic view illustrated in Figure 6.

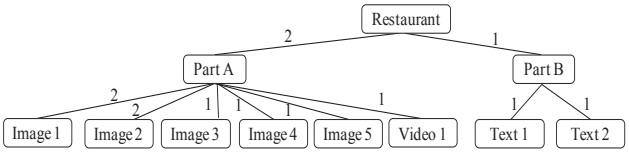


Fig. 6 A generic view Gp constructed from Figure 3(c) and figure 4(c).

The construction of generic profiles from several specific profiles, and the generic views from several specific views helps to simplify and accelerate the adaptation process. So, the Services Management don't need to calculate the distance between all the specific profiles and the user's context, but it selects, first, a relevant generic profile and view, then, it searches a relevant specific profile and view that correspond to the user's context from the specific profiles and views attached to the relevant generic profile and view. In the next paragraph, we explain how the generic profiles are used for selecting specific views.

V. USING GENERIC PROFILES FOR SELECTING SPECIFIC VIEWS

Let consider that many specific views can be specified by a Web Service provider. Each specific view is linked to a particular profile. Hence, from a user profile, the goal of adaptation is to find the specific view that satisfies most of the target device constraints. That is, compute a distance between the user profile and the Web Service profile. An Adapted solution is thus the specific profile (associated with the specific view) that is close to the user profile.

For that purpose, since the user profile and the Web Service profile have the same structure, we have proposed the following distance between profiles:

$$\text{Sim}(Tcu, Tcs) = \left[\sum_{i=1}^{N} DC(Ecui, Ecsi) \times (W(Ecui) + \beta)\right] \Big/ N$$

Tcu/Tcs: tree of user and service context

Ecui/Ecsi: element i of the service/user context

W(Ecui): weight of Epui element of the user context

N: number of the context elements

$$DC(Epui, Epsi) = \begin{cases} 1 & \text{if } Epui = Epsi \\ 0 & \text{else} \end{cases}$$

In this equation, we compare the content of each element of the user's context (Ecui) with the same element of the service context Ecsi calculating a degree of correspondence between the two (DC). If both have the same content, then DC takes the Value 1, otherwise 0. For example, if the user uses an operating system such as Windows, while the service works with a Unix system, the two systems are different, their DC is equal to 0. To take into account the degree of importance of the information associated to the user's context, we must multiply DC with W(Ecui). But sometimes the user does not provide weights for all preferences in this case and to prevent it assumes 0 ß = 0.01.

However, from a particular user profile it is not suitable to compute a distance between all the specific profiles. Actually, the frameworks presented in the Related Work

section explore individually each service description and compute such kind of distance in order to select the relevant services. In many cases, this situation could overload the system. First, because many specific profiles could have been specified. Second, because the user profile may vary a lot of times. Hence, many comparisons have to be computed and updated.

Consequently, we propose to group some specific profiles and views into the generic structures presented in the previous section. Groups can be formed for particular devices, particular characteristics. . . Moreover, groups can be formed according to the distances between specific profiles. More precisely, if two profiles are quite similar thanks to the above defined distance, it means that they could be grouped into a generic structure.

Thanks to generic profiles and views, we can search more efficiently an adapted solution by: (1) searching a generic web service profile that comply most of the user profile and (2) from the selected generic profile, searching a corresponding specific web service that comply most of the user profile. It is obvious that this search complexity is less than exploring individually all the specific possibilities.

VI. USING GENERIC VIEWS FOR GENERATING SPECIFIC VIEWS

Let consider that it might be possible that no specific views can be executed from a user profile. For instance, all specific profiles may have a low distance with the user profile, even if the distance from the generic profile is better. This means that from a generic view it is possible to generate a new specific view better than the existing ones.

Example 3. A user which is using his iPhone may prefer that the service content is in the form of text and video only.

Turn back to the Figures 3 and 4, we note that there isn't specific view relevant with the previous user profile (neither *Sv1* nor *Sv2*).

- The first specific view of Figure 3 (c) (i.e., *Sv1*) is relevant with the user that prefers a service displaying images and videos only.
- The second specific view of Figure 4 (c) (i.e. *Sv2*) is relevant with the user that prefers a service presenting images and texts only.

In order to solve such a situation, we have defined the following procedure for generating specific views from a generic view:

For each element of the generic view, if it complies with the user profile, instantiate it, otherwise do not consider it. Hence, many adapted solutions can be computed from a generic view. The solutions that contain most of objects that comply with the targeted profile are privileged solutions. Hence, a score is computed accordingly. Figure 7 presents the specific view generated by our procedure from the generic view presented in Figure 6.

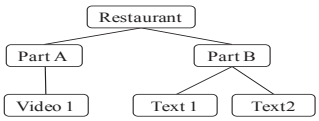


Fig. 7 Specific view generated (Sv3)

In many cases, when no specific view corresponds to the current user profile, the frameworks presented in the Related Work section provide no solution. Our proposal ensures to the user the usability of Web services at anytime even if no alternative has been specified in advance according to his context.

VII. IMPLEMENTATION

A Java-based prototype has been implemented to validate our proposal. In this prototype, we handle many Web Services related to restaurants, weather forecast, traffic information… In Figure 8, we present a Web Service providing information about a restaurant. More precisely, we illustrate in the figure one specific view when executing this Web Service. Each time a new service is imported into the system, its service description is added to the registery and the generic profiles and views are updated if needed.

Fig. 8 A Web Service presenting a restaurant

Consider now a user that uses an iPhone and that prefers contents in the form of texts and images exclusively. All these contraints are specified in his profile. In such a case, the specific view presented in Figure 8 cannot be executed correctly on his device (because it contains videos that are not supported). Our prototype will first select a generic profile close to the user profile, thanks to the distance detailed in Section 5. Then, it executes the relevant specific view associated to such a generic profile. Since, a specific view complies with the user profile, this one is directly presented to the user. Figure 9 illustrates the adapted result, note that now no videos are presented and that the user could continue using the Web service with his handled device.

Fig. 9 Result of service adapted to iPhone

In this first example, an alternative specific view that corresponds to the user profile has already been designed beforehand. However, thanks to generic profiles, we do not search in the set of all the specific service profiles. In fact, by selecting a relevant generic profile, we search only specific profiles that are potentially good candidates.

Of course, if the user profile is changing, e.g., the user is now using a laptop and prefers contents in the form of videos and images, our CA-WIS platform detects on-the-fly and transparently such evolution and, if the previous displayed specific view does not comply with the current user profile, it displays another relevant specific view as

shown in Figure 10 by using the same process.

Fig. 10 Result of service adapted to the laptop

Suppose now that the service providers have designed no alternative beforehand for the following situation: the user uses his laptop and prefers that contents must be in the form of texts and video. In contrast with the related work, our platform selects a generic profile but won't find a specific service profile that complies with the user profile. In such a case, based on the generic view our platform will compute a new specific view that does not already exist thanks to the method presented in Section 6. Concretely, this method will preserve as much as possible the information designed by service providers, instead of transforming and degrading the contents. Figure 11 presents the new computed specific view. Note that it merges the texts presented in Figure 9 and the videos presented in Figure 10.

Fig. 11 Result of service adapted to the iPhone

As you can see, our CA-WIS platform is able (1) to select efficiently specific views thanks to generic service profiles and (2) to produce new alternatives that has not been designed beforehand by service providers, thus ensuring to users the continuity and the usability of Web Services at anytime, anyhow and anywhere.

VIII. CONCLUSIONS

In this article, we proposed a new architecture that allows adapting automatically a service to changes in the user's context.

To achieve the adaptation process in this architecture, we focus on a new method. This method aims to group some specific profiles and views into generic profiles and views, in order to accelerate the selection of relevant Web Service.

Furthermore, if no specific views can be executed from a user profile, based on generic views, we have proposed to generate new specific views to ensure users to still interact with desired services. Finally, our proposal has been tested through our CA-WIS platform.

REFERENCES

[1] M. Cremene, M. Riveill, C, Martel, C. Loghin and C. Miron, "Adaptation dynamique de services". In DECOR'04, Grenoble, France, 2004.

[2] O. Fouial, "Découverte et fourniture de services adaptatifs dans les environnements mobiles". PHD, Paris, 2004.

[3] Z. Kazi-Aoul, I. Demeure and J.C. Moissiniac, "Une architecture générique pour la fourniture de services multimédia adaptables illustration par un scénario". In Ubimob'04, June 2004.

[4] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M.H. Butler, and L. Tran, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0". In W3C, 2001.

[5] T. Melliti, "Interopérabilité des Services Web complexes. Application aux systems multi-agents". PHD, Paris IX Dauphine University, 2004.

[6] A. Pashtan, A. Heusser and P. Scheuermann, "Personal Service Areas for Mobile Web Applications". In IEEE Internet Computing archive, volume 8, n° 6, p. 34-39, USA, 2004.

[7] B. Soukkarieh, F. Sedes, "Dynamic services adaptation to the user's context". In Proceedings of the Fourth International Conference on Internet and Web Applications and Services (ICIW), p.2236228, IEEE Computer Society 2009.

[8] *B.* Soukkarieh, "L'adaptation au contexte dans un système d'information web. Plateforme CA-WIS". In Ingénierie des Systèmes d'Information 14(1), p.91-116, 2009.

[9] SOAP, W3C, Recommandation W3C 24 Juin 2003. The last version: http://www.w3.org/TR/soap12-part0/ (2006).

[10] UDDI, W3Schools, http://www.w3schools.com/wsdl/wsdl_uddi.asp (2006).

[11] WSDL, W3C Note 15 March 2001, the last version: http://www.w3.org/TR/wsdl. (2006).

Sébastien Laborie is an associate professor at the Computer Science Laboratory of the University of Pau (LIUPPA - France) in the T2i team. He obtained his Ph.D. degree in Computer Science from University Joseph Fourier at Grenoble (France) in 2008. His research topics are at the crossroads of multimedia document specification and adaptation, spatio-temporal quantitative and qualitative representation and reasoning, user profile modeling and Semantic Web technologies.

Bouchra Soukkarieh is a computer engineer in the IRIT laboratory of the University Paul Sabatier. She obtained her Ph.D. degree in Computer Science from University Paul Sabatier at Toulouse (France) in 2010. Her research topics are at the context-aware adaptation, user profile modeling, Web Services and Web Information system.

Pr. Florence Sèdes is a Professor at the University of Toulouse, Paul Sabatier University in France. She is deputy director of the IRIT, Institute of Resarch in Informatics of Toulouse. She is also head of the national research network GDR i3 (Information Interacting Intelligence) of the French CNRS. Her main research topics concern information systems and databases, with applications dedicated to multimedia, metadata and SoLoMo (Social, Localization, Mobility) in ambient intelligence.